Hauptseminar

RoboCup

Thema

Die C_OMR_OS - Architektur

Zusammenfassung

In dieser Arbeit wird die Software - Architektur der CoPS Stuttgart behandelt. Sie stellt ein Multi-Agenten-System aus Elementaragenten dar. Ein weiteres Thema sind Verhaltensmuster der Roboter beim Fussballspielen.

Wintersemester 2000 / 2001

Universität Stuttgart
Fakultät Informatik
IPVR

Name
Jörg Rüdenauer

Betreuer
Dipl.-Inf. M. Schulé
Prof. Dr. P. Levi

Inhaltsverzeichnis

1 Einleitung und Überblick

Die RoboCup - Inititative[ROB] wird immer mehr zum Standard - Szenario beim Vergleich der neuesten Entwicklungen auf dem Gebiet der Verteilten Künstlichen Intelligenz und der Robotik. Diese Arbeit behandelt die Software - Architektur des Stuttgarter RoboCup - Teams Cooperative sOccer Playing Robots Stuttgart (CoPS), die auf der an der Universität Stuttgart entwickelten C_OMR_OS (Cooperative Mobile Robots Stuttgart) - Architektur basiert.

Die C_OMR_OS - Architektur ist ein generischer Ansatz zur Steuerung und Koordination einer Gruppe von mobilen Robotern. Sie ist als sog. Multi - Agenten - System (MAS) konzipiert, das aus Elementaragenten besteht.

In Abschnitt 2 gebe ich daher zunächst eine Einführung in den Aufbau dieser Elementaragenten. Ihr Zusammenspiel im Rahmen des MAS ist Thema von Abschnitt 3. In Abschnitt 4 folgt eine detaillierte Beschreibung der CoPS - Architektur. Dabei gehe ich zunächst genauer auf die derzeitige Situation ein und schlage dann einige Änderungen vor, die im Zuge des derzeit ablaufenden Studienprojektes verwirklicht werden sollten. Abschnitt 5 liefert schliesslich einen Einblick in die Verhaltensweisen einzelner und mehrerer kooperierender Roboteragenten beim Fussballspielen.

2 Elementaragenten

2.1 Überblick und Definition

Für den Begriff eines *Agenten* gibt es keine einheitliche Definition. Zu einem Agenten gehört aber sicherlich, dass er selbstständig und autonom ist (d.h. aufgrund vorgegebener Regeln eigene Entscheidungen trifft), Informationen aus seiner Umgebung aufnimmt, sie versteht und auf ihnen basierende Handlungen vornimmt, die seine Umgebung verändern. Untrennbar mit dem Agenten verbunden ist das *Multi - Agenten - System* (MAS), in das er eingebettet ist. Ein MAS ist ein offenes, komplexes System, das aus autonomen Agenten aufgebaut ist, die untereinander interagieren. Häufig kann das MAS selbst wieder als Agent in einem größeren System angesehen werden; z.B. ist ein einzelner Roboter der CoPS als MAS modelliert, ist aber innerhalb der Mannschaft als einzelner Agent zu betrachten.[SNZ99]

Die Modellierung als MAS sichert nicht nur Lokalität und Informationskapselung, sie führt im Idealfall auch zu hoher Wiederverwendbarkeit, Fehlertoleranz und dynamischer Konfigurierbarkeit durch die unterschiedliche Kombination und Kooperation verschiedener Agenten [LBLM98].

Der im Rahmen von C_OMR_OS entwickelte *Elementaragent* (EA) besteht aus einem Agentenkopf und einem Agentenkörper (s. Abb. 1). Er repräsentiert eine abstrakte

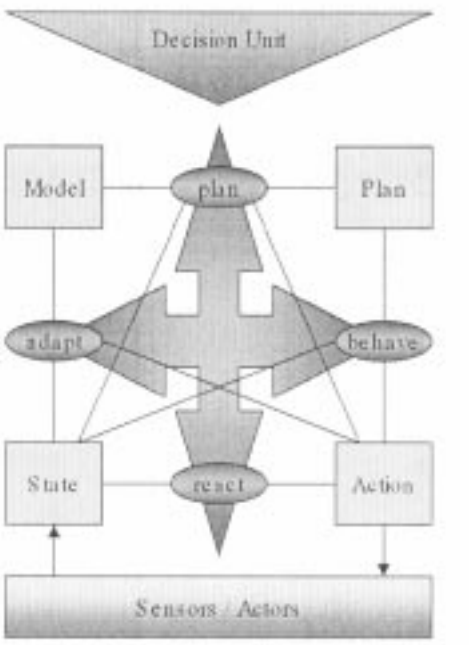

Abbildung 1: Aufbau eines Elementar - Agenten (aus [LOSL00])

Regelschleife, die Informationen von Sensoren aufnimmt und Befehle an Aktoren weitergibt. Sowohl Sensoren als auch Aktoren können Entsprechungen in der realen Welt haben, z.B. Laserscanner oder Elektromotoren. Im Allgemeinen sind aber virtuelle Sensoren / Aktoren gemeint – andere Elementaragenten. Sensoren und Aktoren zusammen definieren das *autonome Gebiet* des EA.[LBLM98]

Der Elementaragent kann verschiedenste Formen haben (s.u.). Im einfachsten Fall besteht er aber nur aus einer (nicht terminierenden) Schleife, die einige Variablen verändert.

2.2 Agentenkörper

Der Agentenkörper enthält die eigentliche Funktionaliät des EA. Er besteht sowohl aus Wissen (Rechtecke in Abb. 1) als auch aus Handlungszyklen, die auf es zugreifen und evtl. ändern (Ovale). Das Wissen lässt sich in vier Teilkomponenten zerlegen:

- *Model* beinhaltet das Wissen des Agenten über seine Umgebung. Im einfachsten Fall sind dies einige Variablen, in High-Level - Agenten können alle Formen moderner Wissensrepräsentation wie z.B. semantische Netze zur Anwendung kommen.
- *Plan* steht für die Pläne des Agenten, die sein "Herzstück" darstellen. Sie repräsentieren sog. prozedurales Wissen des Agenten, also das Wissen, wie gewisse Probleme gelöst werden können. Durch die Ausführung der Pläne entsteht die Interaktion des EA mit seiner Umwelt. Die einfachsten Pläne sind als Prozeduren implementierbar.
- *State* ist der momentane Zustand, in dem sich der EA befindet. In realen Programmen ist er selten explizit modelliert; nichtsdestotrotz ist er für das theoretische Verständnis des Elementaragenten essentiell.
- *Action* sind die Aktionen, die der Agent ergreifen kann. Sie hängen gänzlich von den Aktoren ab, auf die er Zugriff hat.

Die Handlungszyklen sind von den Plänen bestimmt. Sie werden im Agentenkörper als Programmschleife ausgeführt und lassen sich in folgende Kategorien einteilen:

1. *react* (reagieren) ist die einfachste Form des Zyklus. Er besteht aus festen Kontrollparametern und einem festen Kontrollalgorithmus und stellt eine einfache Regelschleife dar. *react* ist aufgrund seiner Einfachheit der mit Abstand am Häufigsten eingesetzte Zyklus.
2. *adapt* (anpassen) kann die Kontrollparameter ändern, hat also Einfluss auf das Modell, wo diese gespeichert sind. Somit wird der Agent lernfähig. So könnte z.B. ein Roboter, der mehrfach zu spät abgebremst hat und daher mit anderen Robotern kollidierte, in Zukunft schon früher bremsen.
3. *behave* (verhalten) kann seinen Algorithmus ändern, in dem er vorgefertigte Planskelette dafür heranzieht. Die Entscheidung, welches Planskelett ausgewählt werden soll, sowie das Erweitern des Skeletts zu einem vollständigen Plan geschieht mit Hilfe des *Model* und der Kontrollparameter. Die CoPS - Roboter verwenden diese Art des Zyklus auf der taktischen Ebene (s. Abschnitte 4 und 5).
4. *plan* (planen) stellt die höchste Ebene der Handlungszyklen dar. Ihm ist es möglich, neue Pläne, also neue Kontrollalgorithmen, zu entwickeln und damit den Agenten tiefgreifend zu verändern.

Ein EA kann mehrere Handlungszyklen haben, die sich wiederum einer oder mehreren der obigen Kategorien zuordnen lassen.[LOSL00], [SNZ99]

Heutzutage sind *adapt* - oder *behave* - Zyklen aufgrund der höheren Komplexität noch eher selten. Ein Agent, der *plan* - Zyklen ausführen kann, beinhaltet schon eine große Menge "Künstlicher Intelligenz". Ein Fussballroboter dieser Stufe könnte z.B. von sich aus lernen, den Torwart zu umspielen, falls Distanzschüsse keinen Erfolg bringen.

2.3 Agentenkopf

Der Agentenkopf als Entscheidungseinheit steuert und kontrolliert den Agentenkörper, indem er die dort ausgeführten Pläne überwacht. Er kann die Ausführung anhalten bzw. fortsetzen und ist zuständig für die Zuteilung von Ressourcen (z.B. Rechenzeit) innerhalb des Agenten. Dafür muss jeder Plan Informationen über die Ressourcen bereitstellen, die er benötigt. Ausserdem repräentiert der Kopf die Fähigkeiten des Agenten nach aussen hin und kann dadurch den Agenten eine Rolle in einem Entscheidungsnetzwerk übernehmen lassen (s. Abschnitt 3.2). Die Fähigkeiten des Agenten werden durch seine Pläne festgelegt, mit denen eine Rolle zur Laufzeit stets in Verbindung ist [LBLM98], [LOSL00].

In der CoPS - Architektur (s. Abschnitt 4) hat der Agentenkopf nur recht wenig Funktionalität, da viele seiner Fähigkeiten nicht benötigt werden. Daher ist er auch in der Implementierung nicht vom Agentenkörper getrennt.

3 Kooperation zwischen Elementaragenten

Wie schon erwähnt, agieren Elementaragenten stets in einem Multi - Agenten - System (MAS). Dabei ist eine Kooperation zwischen ihnen unerlässlich. In der C_OMR_OS - Architektur sind Agenten durch zwei Arten von Netzwerken verbunden:

- Die Agentenkörper durch Datenflussnetzwerke, über die Modelldaten durch (virtuelle) Sensor - Aktor - Kanäle fliessen
- Die Agentenköpfe durch Entscheidungsnetzwerke, die die Koordination der einzelnen Agenten übernehmen und so die effektive Problemlösung ermöglichen

3.1 Verbindungen zwischen Agentenkörpern

Die Körper der Elementaragenten sind über ein Datenflussnetzwerk (DFN) verbunden. Dabei sind Eingänge eines Agenten als virtuelle Sensoren zu betrachten, Ausgänge können als virtuelle Aktoren angesehen werden. Zwischen zwei Agenten besteht somit eine Client - Server - Verbindung: Der Client schickt Kontrollanweisungen und Modellinformationen durch seinen Aktorkanal an den Server und erhält auf seinem Sensorkanal Statusinformationen und Sensordaten zurück. Das DFN stellt permanente Verbindungen der Agenten dar; seine Struktur wird durch die Schnittstellen der Autonomiezyklen bestimmt. Durch das DFN werden die Fähigkeiten und die Organisation des MAS festgelegt [LBLM98]. Beispiele für DFN finden sich u.a. in Bild 2 und Bild 3.

3.2 Verbindungen zwischen Agentenköpfen

Die Köpfe der Elementaragenten nehmen an einem oder mehreren *Entscheidungsnetzwerken* (EN) teil. Ein EN ist zur Bewältigung einer bestimmten Aufgabe des MAS gedacht; für neue Aufgaben wird kein neuer Agent eingeführt, der ihre Erledigung koordiniert, sondern ein neues EN zwischen den bestehenden Agenten aufgebaut. Dabei kann jeder Agent eine oder mehrere *Rollen* im EN übernehmen, falls er dazu fähig ist (bestimmt durch den Agentenkopf). Die Entscheidungsfindung verläuft dann in einzelnen Phasen, wobei ein Phasenübergang von allen Mitgliedern im EN gleichzeitig ausgeführt wird. Dies ist möglich, sobald bei jedem Agenten gewisse Restriktionen erfüllt sind. Ist das nicht der Fall, so versucht er zunächst, seinen eigenen Zustand zu ändern. Gelingt dies nicht, so teilt er es den anderen Agenten mit, und eine gemeinsame Lösung wird gesucht. Wird auch eine solche nicht gefunden, können die Restriktionen gelockert werden [LBLM98].

Meiner Ansicht nach stellen Entscheidungsnetzwerke einen hochinteressanten Ansatz dar, in dem enormes Potential steckt. Sie bedürfen jedoch recht hoch entwickelter Agenten mit grosser Flexibilität, um es voll auszuschöpfen; denn dies wird nur erreicht, wenn Agenten in mehreren verschiedenen EN teilnehmen und selber über ihre Teilnahme entscheiden. Zudem ist die Kommunikation zwischen Robotern noch relativ aufwändig und langsam, so dass Verhandlungen nur eingeschränkt möglich sind.

Deshalb verwendet die CoPS - Architektur noch keine Entscheidungsnetzwerke.

3.3 Beispiel: Formationsfahren

Zum besseren Verständnis der einzelnen Netzwerke zeigt Abb. 2 das DFN eines Roboters (rechts) und das EN für die Aufgabe "Formationsfahren", bei der mehrere Roboter eine gewisse relative Position zueinander einhalten sollen (links). Diese Aufgabe ist besonders gut zur Demonstration der C_OMR_OS - Architektur geeignet, da sie ein hohes Mass an Koordination und Kooperation zum Einen zwischen den einzelnen Robotern, zum Anderen zwischen den Agenten innerhalb eines Roboters erfordert.

Im DFN sieht man z.B., dass der Pilot vom Navigator über dessen Aktorkanal die vorgesehene Trajektorie bekommt und im Gegenzug an dessen Sensorkanal Informationen über den tatsächlich gefahrenen Weg zurückliefert. In gleicher Weise sendet der Pilot Fahrbefehle an die Roboterkontrolle und bekommt Odometrie - Daten zurück. Der Virtuelle Abstandssensor vereinigt die Daten des Laserscanners und der Ultraschallsensoren und fungiert so als allgemeiner Sensor gegenüber der Hinderniserkennung.

Das EN besteht aus folgenden Phasen:

1. "Planen der Trajektorie der gesamten Formation": Hier wird die globale Richtung bestimmt, in die die Formation als Ganzes sich bewegen soll

2. "Aushandeln der topologischen Positionen": Nun wird bestimmt, welcher Roboter welche Position innerhalb der Formation einnehmen soll

3. "Aushandeln der einzelnen Trajektorien": Jetzt kann ermittelt werden, in welche Richtung sich jeder einzelne Roboter bewegen muss

4. "Fahren": Schliesslich fahren die Roboter in die vorher bestimmte Richtung.

Falls eine Phase fehlschlägt, geschieht ein *backtracking*, also eine Rückkehr zur vorhergehenden Phase. Der Navigator übernimmt die Rolle "Planendes Mitglied" während der ersten und die Rolle "Passives Mitglied" während der zweiten Phase.

Obwohl die Symbolik in Abb. 2 nahezu gleich ist, ist es wichtig, dass das Entscheidungsnetzwerk keinen eigenen Agenten darstellt, sondern nur die Kommunikationsstruktur zwischen den Köpfen der Elementaragenten, die an ihm partizipieren.

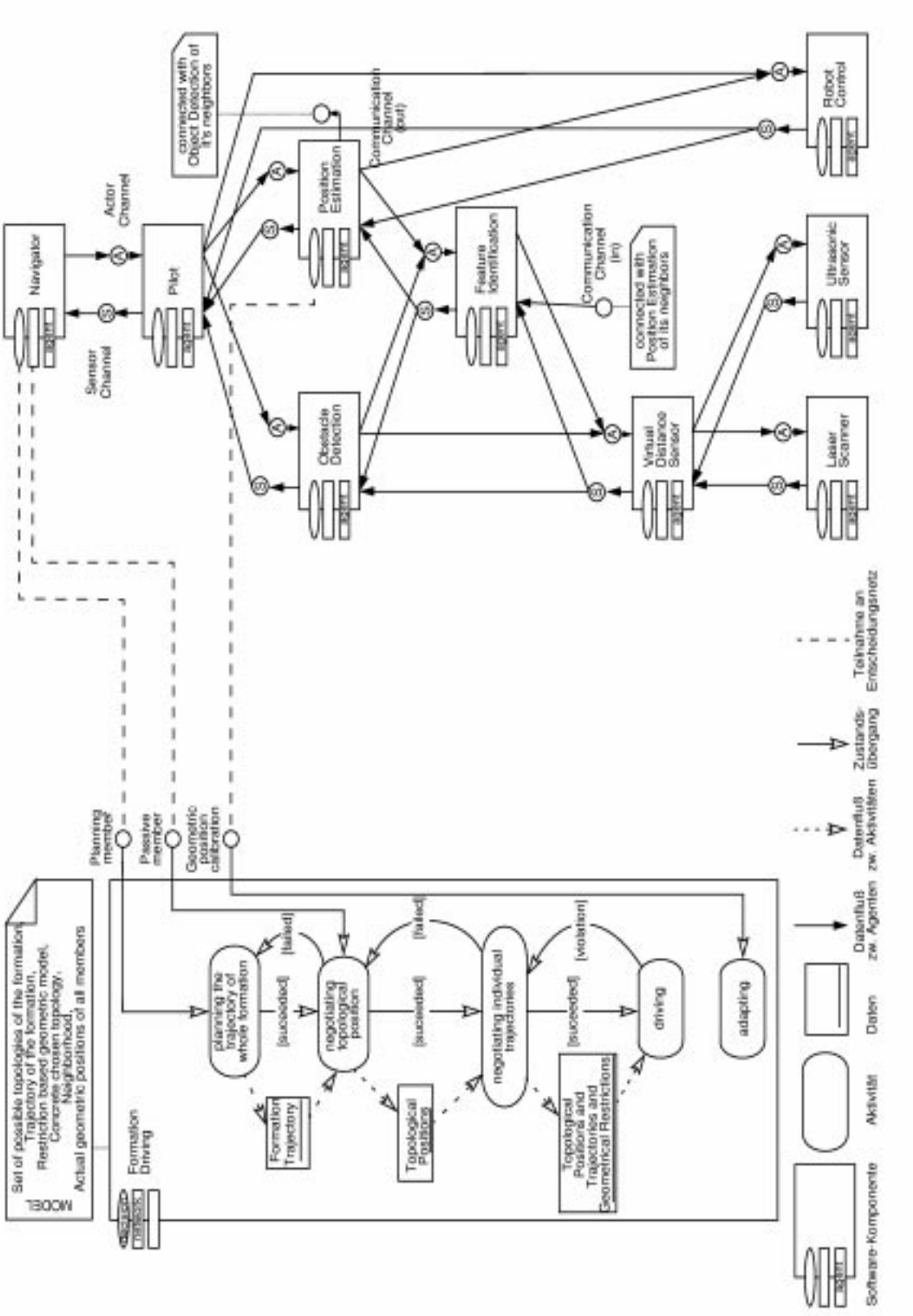

Abbildung 2: Netzwerke zum Formationsfahren, in UML (aus [LBLM98])

4 Die CoPS - Architektur

Mit den CoPS (Cooperative Soccer Playing Robots Stuttgart) nimmt die Universität Stuttgart seit 1999 erfolgreich am RoboCup teil. Das RoboCup - Szenario wurde deshalb zur Demonstration der praktischen Anwendbarkeit der C_OMR_OS - Architektur gewählt, weil es eine standardisierte, klar abgrenzte Aufgabe darstellt. Trotzdem ist die Aufgabe, im Team Fussball zu spielen, hinreichend komplex, so dass neben rein reflexiven Fähigkeiten der Roboter auch Planung eine erhebliche Rolle spielt – sowohl für den einzelnen Roboter als auch kooperative Planung im gesamten Team.

Gemäß der C_OMR_OS - Architektur sind die CoPS als Multi - Agenten - System entworfen, wobei einerseits die einzelnen Roboter als Agenten betrachtet werden können, die ein Team bilden, andererseits auch jeder Roboter in sich wieder ein MAS darstellt, das im Folgenden näher erläutert werden soll.

4.1 Derzeitiger Aufbau innerhalb eines Roboters

Die Software jedes Roboters besteht aus einzelnen Agenten, die miteinander kommunizieren und so ein MAS bilden. Die Agenten lassen sich grob in drei Ebenen aufteilen: Die reflexive, die taktische und die strategische Ebene. Jede Ebene hat andere Anforderungen an die Reaktionszeit: Während es genügt, strategische Entscheidungen alle paar Sekunden zu treffen, sollte über taktische Maßnahmen in Sekundenbruchteilen entschieden werden, und die Zykluszeit auf der reflexiven Ebene muss im Millisekundenbereich liegen, um z.B. sich bewegende Hindernisse zu vermeiden oder den rollenden Ball zu verfolgen.

Weiter sind aus Effizienzgründen jeweils mehrere Agenten in einem Thread zusammengefasst. Die bisherige Aufteilung verdeutlicht Abb. 3. Ein kleines Rechteck steht jeweils für einen Agenten, wobei die Agenten, die von einer Graustufe liegen, jeweils in einem Thread laufen. In der Implementierung sind diese im Augenblick auch in je einem Modul zusammengefasst.

4.1.1 Strategische Ebene

Auf der strategischen Ebene befindet sich nur ein Agent, der Stratege. Er tritt bei selten vorkommenden Ereignissen (Tor, Rote Karte, ...) in Aktion und entscheidet über das globale Verhalten des Roboters, z.B. über das Risiko der Aktionen oder darüber, ob die Spielweise eher defensiv oder eher offensiv ausgerichtet sein soll. Daher kommuniziert er mit den beiden Agenten auf der taktischen Ebene. Ausserdem kann er mit den Strategen anderer Roboter konferieren, da die strategischen Entscheidungen unweigerlich zwischen den Teammitgliedern abgesprochen sein müssen.

Derzeit ist die Funktionalität des Strategen noch sehr eingeschränkt und deshalb ist auch noch kein eigenes Modul für ihn vorhanden.

4.1.2 Taktische Ebene

Die taktische Ebene besteht aus zwei Agenten: der Szenenerkennung und dem Navigator. Die Szenenerkennung sammelt die Daten der (virtuellen) Sensoren auf der reflexiven Ebene, verarbeitet sie und stellt das Ergebnis dem Navigator bereit. Zum Beispiel berechnet sie aus der Position des Balles relativ zum Roboter, die von der Objekterkennung

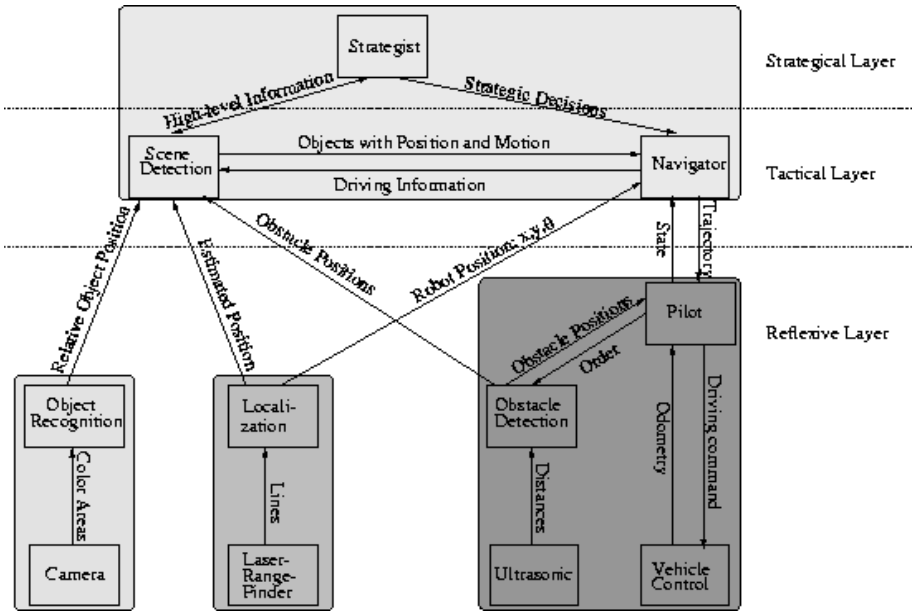

Abbildung 3: Architektur eines Roboters (aus [LOSL00])

geliefert wird, und der absoluten Position des Roboters, die sie von der Lokalisierung bekommt, die absolute Position des Balles auf dem Spielfeld. Die Szenenerkennung kommuniziert auch mit der Szenenerkennung anderer Teammitglieder. Dadurch wird zum Einen das eigene Wissen vermehrt: z.B. könnte der Ball ausserhalb des Blickfeldes des Roboters sein, aber von anderen Robotern gesehen werden. Zum Anderen werden gewisse Schlussfolgerungen, u.a. die Unterscheidung zwischen eigenen und gegnerischen Robotern, überhaupt erst durch die Kommunikation zwischen den Teammitgliedern ermöglicht. Schliesslich kann mit Hilfe der Daten anderer Roboter auch die Plausibilität der eigenen Daten, z.B. der geschätzten Position auf dem Spielfeld, überprüft werden.

Im Augenblick sind die Fähigkeiten der Szenenerkennung noch recht rudimentär und bestehen hauptsächlich in einem Zusammenführen der Daten des eigenen Roboters und der über das Netzwerk empfangenen Daten der anderen Teammitglieder. Die Kommunikation wird mit CORBA realisiert, das eine einfache, standardisierte Möglichkeit des Nachrichtenaustauschs bereitstellt.

Der Navigator fordert von der Szenenerkennung die Daten über die aktuelle Situation an und entscheidet sich dann anhand dieser Daten und der aktuellen Strategie für ein bestimmtes Verhalten (s. auch Abschnitt 5). Zu jedem Verhalten gehört ein Planskelett (s. Abschnitt 2), das mit den aktuellen Daten zu einem Plan vervollständigt wird. Dieser berechnet dann eine Wegvorgabe (Trajektorie), die dem Piloten als Befehl übergeben wird.

4.1.3 Reflexive Ebene

Die Reaktionszeiten auf der reflexiven Ebene müssen um Einiges kürzer sein als die auf der taktischen Ebene, wo es ausreicht, wenn Entscheidungen innerhalb von Zehntelsekunden getroffen werden. Daher finden sich auf der reflexiven Ebene auch mehr Threads wieder. So kann dadurch, dass die Agenten, die die physikalischen Sensoren (Kamera,

Laser) auswerten, in einem eigenen Thread ablaufen, äusserst rasch auf sich verändernde Umgebungen reagiert werden.

Einige der Elementaragenten dieser Ebene korrespondieren direkt zu Sensoren / Aktoren der Hardware. Dies sind die Kamera, der Laserabstandsmesser, der Ultraschall und die Fahrzeugkontrolle. Diese überwacht Geschwindigkeit und Beschleunigung der Räder und löst den Mechanismus zum Schiesssen des Balles aus.

Die Kamera liefert Farbbereiche an die Objekterkennung, die anhand ausgefeilter Algorithmen innerhalb kürzester Zeit (40 ms) die Position von Objekten wie Ball oder Tore bestimmen kann. Der Laser übermittelt ein Linienmuster an die Lokalisierung, die daraus die Position des Roboters auf dem Spielfeld schätzt. Anhand der Ultraschallsensoren kann die Entfernung zu Hindernissen abgeschätzt werden. Diese Aufgabe nimmt die Hindernisentdeckung wahr. Objekterkennung, Lokalisierung und Hindernisentdeckung liefern ihre Daten jeweils an die Szenenerkennung.

Die entdeckten Hindernisse werden ausserdem direkt dem Piloten zur Verfügung gestellt, damit dieser sie vermeiden kann. Der Navigator auf der taktischen Ebene gibt dem Piloten eine Trajektorie vor, die dieser den Roboter abfahren lässt, wobei allerdings geringfügige Abweichungen erlaubt sind. Dazu steht der Pilot in ständiger Verbindung mit der Fahrzeugkontrolle, die ihm Odometriedaten zurückliefert.[LOSL00]

Logisch gesehen sollten die Agenten, die auf die Ultraschallsensoren zugreifen, ebenso in einem eigenen Thread enthalten sein wie diejenigen der Kamera und des Lasers. Da jedoch die Fahrzeugkontrolle und die Ultraschallsensoren über die gleiche Hardwareschnittstelle angesprochen werden, ist eine Trennung physikalisch unmöglich und daher auch auf Softwareebene sinnlos.

4.2 Vorgeschlagene Änderungen

Im Rahmen des Studienprojektes "Robocup" wird innerhalb des nächsten halben Jahres eine umfassende Reimplementierung grösserer Teile der Architektur geschehen, hauptsächlich des Navigators und des Piloten. Der Grund dafür ist, dass das bisherige Vorgehen, dem Piloten eine Trajektorie zu übergeben und diese stetig anzupassen, sich als zu unflexibel und zu langsam erwiesen hat. Im Zuge dessen schlage ich einige kleine Änderungen am Multi - Agenten - System vor, die in Abb. 4 verdeutlicht sind.

Zunächst wird die Hindernisentdeckung in Zukunft hauptsächlich den Laser benutzen, da die Ultraschallsensoren häufig ein zu ungenaues Bild der Umgebung liefern (siehe dazu auch [SLE99]). Die Ultraschallsensoren werden nur noch eingesetzt, wenn der Laser aufgrund seiner Beschränkung auf eine bestimmte Höhe nicht einsetzbar ist. Daher sollte der entsprechende Agent auch dem Thread angehören, in dem der Laser und die Lokalisierung angesiedelt sind.

In Zukunft wird der Navigator dem Piloten nicht mehr eine Trajektorie vorschreiben, sondern ihm weitaus abstraktere Fahrbefehle geben, z.B. "Fahre in Schussposition" oder "Bewege den Ball von der Bande weg". Um diese Befehle ausführen zu können, benötigt der Pilot erheblich mehr Wissen über die Situation auf dem Spielfeld. Daher bietet es sich an, die Szenenerkennung als gemeinsamen Speicher zwischen den einzelnen Threads zu benutzen. Die (virtuellen) Sensoren schreiben dort ihre Daten hinein. Der Pilot liest sie auf einer tieferen Ebene wieder aus – z.B. interessieren ihn nur die Position des Balles relativ zum Roboter, nicht die absolute Position. Auf einer höheren Ebene kann die Szenenerkennung erweitert werden und dem Taktiker abstraktere Fragen beantworten,

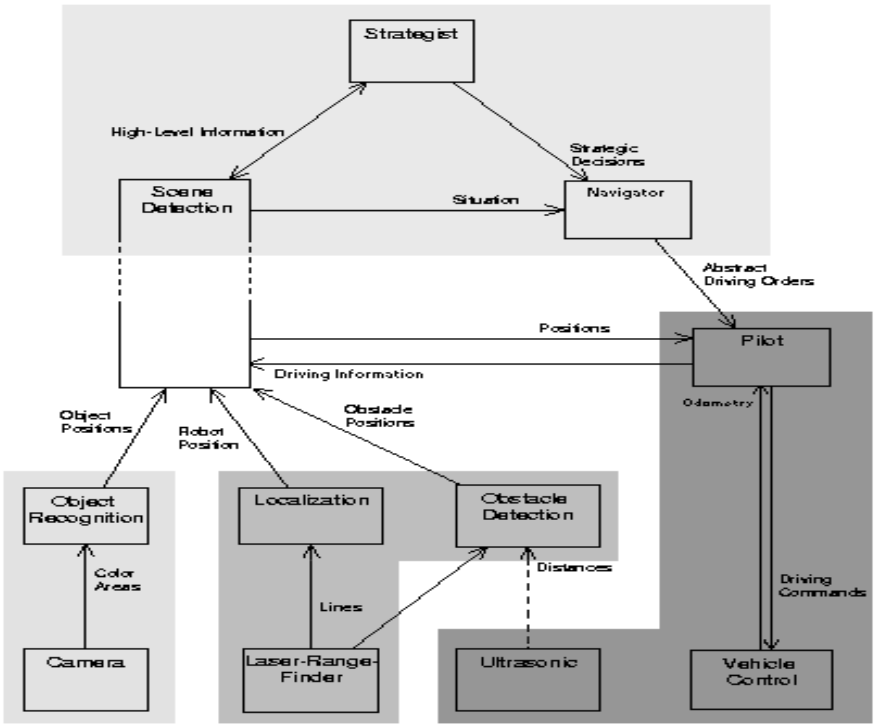

Abbildung 4: Architektur mit den vorgeschlagenen Änderungen

etwa “Welches Team kontrolliert den Ball?” oder “Ist mein Roboter der am nächsten zum Ball stehende unserer Mannschaft?”.

5 Verhaltensmuster

5.1 Verhalten

Für das Fussballspiel ist es auf taktischer Ebene ausgesprochen nützlich, *Verhalten* zu finden, die eine häufig wiederkehrende Anzahl an sequentiellen oder parallelen Aktionen bestimmen. Einige Verhalten einzelner Roboter sind in Tabelle 1 aufgeführt. Die Unterscheidung in defensive oder offensive Verhalten ist eher zur Orientierung gedacht und kann das Verständnis von Rollen (s.u.) erleichtern; sie ist keineswegs in den Verhalten inhärent.

offensiv	**defensiv**
Ball kontrollieren	Ball wegnehmen
Ball suchen	Ball suchen
Ball holen	Ball holen
Bewegung in den freien Raum	Gegnerdeckung
Dribbling mit dem Ball	Ball wegschiessen
Tor schiessen	Ball abblocken

Tabelle 1: Verhalten einzelner Roboter (aus [LOSL00])

Verhalten von Gruppen von Robotern sind etwas komplexer. Für sie ist es notwendig, dass die Agenten auf der taktischen Ebene miteinander verhandeln und so bestimmen, welcher Roboter welche Position einnehmen soll (ähnlich zum Formationsfahren, s. Abschnitt 3.3). Einige mögliche Verhalten von Robotergruppen zeigt Tabelle 2.

offensiv	**defensiv**
Angriffsriegel bilden	Abwehrriegel bilden
Sich freispielen	Pressing
Flügelspiel	Tor verteidigen
Powerplay	Zeitspiel

Tabelle 2: Verhalten von Robotergruppen (aus [LOSL00])

Derzeit werden Gruppenverhalten von den CoPS noch nicht benutzt, da die Navigatoren noch nicht miteinander kommunizieren; mittelfristig ist ein solches Vorgehen jedoch geplant.

5.2 Rollen

Verhalten einzelner Spieler lassen sich wiederum zu *Rollen* zusammenfassen (dies hat nichts mit den Rollen in Entscheidungsnetzwerken zu tun!). Eine Rolle, man könnte sie auch Spielertyp nennen, besteht somit aus einer Untermenge der Verhalten, sowie der Entscheidungskriterien, welches Verhalten wann angewendet werden soll. Derzeit gibt es bei den CoPS die drei Rollen *Verteidiger*, *Mittelfeldspieler* und *Stürmer*. Eine feinere Unterscheidung mit erheblich mehr Rollen ist aber durchaus denkbar und problemlos möglich, da die Verhalten unabhängig von den Rollen definiert sind. Auch ein Wechsel der Rolle eines Roboters während des Spiels kann ohne Weiteres geschehen. So könnte z.B. ein Verteidiger, der den Ball bekommt und freien Weg zum Tor hat, kurzzeitig die

Rolle wechseln und versuchen, ein Tor zu erzielen. Ausserdem können Rollenwechsel vom Strategen vorgenommen werden.

5.3 Strategien

In Zukunft sollen die Strategen der Roboter (s. Abschnitt 4) sich je nach Spielsituation gemeinsam für eine Teamstrategie entscheiden. Eine Strategie besteht, ähnlich wie eine Rolle für einzelne Spieler, aus einer Menge von Gruppenverhalten. Falls unbedingt ein Tor benötigt wird, könnte die Strategie z.B. die Verhalten "Angriffsriegel bilden" und "Powerplay" beinhalten. Jede Strategie fordert ausserdem eine bestimmte Rollenaufteilung, z.B. könnte eine einfache defensive Strategie zwei Verteidiger und einen Mittelfeldspieler benötigen. In Verhandlungen untereinander sollen die Strategen bestimmen, welcher Roboteragent welche Rolle übernimmt. Schliesslich bestimmt die Strategie auch noch, wie riskant die einzelnen Spieler sich verhalten. Ein grösseres Risiko könnte z.B. eine höhere Fahrtgeschwindigkeit oder einen geringeren Sicherheitsabstand zu anderen Robotern bedeuten. Ist eine Mannschaft im Rückstand, so wird sie ein deutlich höheres Risiko eingehen als bei einer Führung.

5.4 Fähigkeiten

Jedes Verhalten bedarf gewisser Fähigkeiten, die der Roboter haben muss, der es anwenden soll. So ist es zum Beispiel für ein Dribbling mit Torschuss erforderlich, dass der Roboter den Ball führen und schiessen kann. Die Fähigkeiten der CoPS sind alle gleich, mit der Ausnahme des Torhüters. Aufgrund seiner speziellen Funktion ist es häufig erforderlich, dass er sich seitlich vor dem Tor hin- und herbewegt. Deshalb ist bei ihm der Schussmechanismus seitlich zur Fahrtrichtung montiert. Auch hat er drei Kameras, um neben das Tor sehen zu können, ohne sich zu drehen.

Durch die Uniformität der Feldspieler ist keinem von ihnen ein bestimmtes Verhalten vorgeschrieben. Daher kann jeder viele verschiedene Rollen annehmen und die Strategie auch während dem Spiel einfach geändert werden. Ausserdem sind sie so auch auf Hardwareseite erheblich einfacher zu warten. [LOSL00]

6 Fazit und Ausblick

Die Architektur der CoPS, die auf der C_OMR_OS - Architektur basiert, stellt ein Multi-Agenten-System dar, das aus Elementaragenten zusammengesetzt ist. In dieser Ausarbeitung habe ich versucht, sowohl die theoretischen Grundlagen dieser Architektur zu vermitteln, als auch einen Einblick in das Zusammenspiel der Agenten der CoPS zu geben. Ausserdem habe ich dargelegt, wie das Spiel der Roboter durch Strategien, Rollen und Verhalten bestimmt wird.

Während des Studienprojektes "Robcup" werden wir einige Änderungen an der Architektur vornehmen, die sie flexibler und die Roboter schneller machen soll (s. Abschnitt 4.2). Später soll auch der Stratege detailliert werden und seine volle Funktionalität erhalten. Ebenso ist an eine erweiterte Kommunikation auf taktischer Ebene gedacht.

Arbeiten auf theoretischem Gebiet beschäftigen sich mit der Modellierung von Entscheidungsnetzwerken in Form von Petri-Netzen und deren halbautomatische Umsetzung in eine Klassenstruktur [BML97] und mit der mathematischen Berechnung von Multi-Agenten-Systemen mit Hilfe der Chaos - Theorie [ABL+98].

Die C_OMR_OS - Architektur hat sich inzwischen in der Praxis mehrfach bewährt, und nichts spricht dagegen, dass sie auch in Zukunft eine wichtige Rolle bei der Steuerung autonomer Robotergruppen spielen wird.

Referenzen

[ABL+98] V.Avrutin, M.Becht, R.Lafrenz, P.Levi, M.Muscholl und M.Schanz. *CoMRoS - Eine Multi-Agenten Architektur für intelligente Robotersysteme.* In Proceedings of the KI 98, Bremen, 1998.

[BML97] M.Becht, M.Muscholl, and P.Levi. *Ein Framework für Kooperationsverfahren zwischen Roboteragenten.* In P.Levi, Th. Bräunl, and N. Oswald, editors, Autonome Mobile Systeme 1997, Informatik aktuell, Berlin u.a., Springer.

[LOSL00] R.Lafrenz, N.Oswald, M.Schulé, and P.Levi. *A Cooperative Architecture to Control Multi-Agent based Robots.* The Sixth Pacific Rim International Conference on Artificial Intelligence (PRICAI-2000), 2000.

[LBLM98] P.Levi, M.Becht, R.Lafrenz, and M.Muscholl. *COMROS - A Multi-Agent Robot Architecture.* In R. Dillmann et al. (eds.), Distributed Autonomous Robotic Systems 3, Springer-Verlag, 1998.

[SNZ99] M.Schanz. *Einführung in die Verteilte Künstliche Intelligenz.* Skript zur Vorlesung, Version 1.1, 1999.

[SLE99] M.Schulé. *Robotik I.* Skript zur Vorlesung, Stand Wintersemester 1999 / 2000.

[ROB] Die RoboCup - Initiative: www.robocup.org